TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Ville Törhönen

Designing a Software-Defined Datacenter

Master of Science Thesis

Examiner: Professor Jarmo Harju
Examiner and topic approved by the Faculty of Computing and Electrical Engineering on December 4th 2013

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
Ville Törhönen: Designing a Software-Defined Datacenter
Master of Science Thesis, 49 pages, 5 Appendix pages
May 2014
Major: Communication Networks and Protocols
Examiner: Professor Jarmo Harju
Keywords: Cloud computing, datacenter, API, SDDC, orchestration, continuous delivery, DevOps, infrastructure provisioning, virtualization

A datacenter is a complex environment, which consists of network equipment, server hardware and storage systems. Traditionally these are managed by system administrators or infrastructure engineers, either manually or by scripting. Cloud computing has pushed this traditional model more into automation-based solutions. Servers and services are expected to be provisioned as fast as possible, usually within minutes.

This thesis examines how a datacenter can be designed to enable fast, reliable and scalable automation. It also has to provide certain high-availability and scalability features, such as virtual machine migration. A multi-vendor environment is expected, but all components must support centralized management. The infrastructure has to provide different APIs (Application Programming Interface) in all components of the datacenter. A datacenter which can be controlled by utilizing these APIs can be called as a *software-defined datacenter*.

An implementation of a software-defined datacenter is presented. The datacenter is based on VMware virtualization, Hewlett-Packard server hardware, NetApp storage with Hewlett-Packard and Cisco networking equipment. A virtualization environment was installed and configured by using features that helped mass-configuration. An orchestration tool was installed, which is the key building block for providing flexible automation workflows. An orchestration workflow was designed to provide customizable virtual machine provisioning, DNS configuration, storage allocation and network configuration.

Additional ideas for improvement are also introduced. These topics include environment upgrades and security patching, IPv6 deployment and automated configuration management. Configuration management is the next step in infrastructure automation, as it enables operating system configuration automation. It is dependent of the infrastructure automation, such as server provisioning, presented in this thesis.

The implementation proves that it is possible to build a software-defined datacenter with multi-vendor hardware. All components, however, must some how support automation or provide an API. The same implementation presented in this thesis could be achieved by different components by different vendors, including hardware and virtualization layer.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
Ville Törhönen: Palvelinkeskuksen ohjelmistopohjaisen hallinnan suunnittelu
Diplomityö, 49 sivua, 5 liitesivua
Toukokuu 2014
Pääaine: Tietoliikenneverkot ja protokollat
Tarkastaja: Prof. Jarmo Harju
Avainsanat: Pilvipalvelut, palvelinkeskus, rajapinnat, automatisointi, jatkuva integraatio, DevOps, virtualisointi

Palvelinkeskukset ovat monimutkaisia ympäristöjä ja ne koostuvat verkkolaitteista, palvelimista sekä levyjärjestelmistä. Tyypillisesti tällaisia laitteita hallitaan manuaalisesti tai erinäisillä skripteillä järjestelmien ylläpitäjien toimesta. Pilvipalvelut ovat kuitenkin ajaneet järjestelmien ohjausta automaatiopohjaisiin ratkaisuihin. Palvelinkeskukselta odotetaan, että virtuaalipalvelimet tulisi pystyä provisioimaan niin nopeasti kuin mahdollista, jopa muutamassa minuutissa.

Tässä työssä tutkitaan miten palvelinkeskus voidaan suunnitella niin, että se mahdollistaa nopean, luotettavan ja skaalautuvan automaation kehittämisen. Tällaisen palvelinkeskuksen on myös mahdollistettava tiettyjä korkean saavutettavuuden ja skaalautuvuuden ominaisuuksia, kuten esimerkiksi käynnissä olevien virtuaalipalvelimien uudelleensijoittamisen virtualisointiympäristössä. Palvelinkeskuksen komponentit tulevat myös erilaisilta toimittajilta, ja niitä on silti pystyttävä hallitsemaan keskitetysti. Palvelinkeskuksen on tarjottava erinäisiä ohjelmallisesti hallittavia rajapintoja hallinnan mahdollistamiseksi. Tällaista palvelinkeskusta voidaan kutsua ohjelmallisesti ohjattavaksi palvelinkeskukseksi (*software-defined datacenter*).

Työssä esitetään toteutus edellä kuvatulle palvelinkeskukselle. Toteutus pohjautuu VMware-virtualisointiin, Hewlett-Packardin palvelinrautaan, NetApp-levyjärjestelmiin sekä Hewlett-Packardin ja Ciscon verkkolaitteisiin. Virtualisointiympäristö asennettiin ja konfiguroitiin käyttämällä massakonfigurointia helpottavia toimintoja. Järjestelmien provisiointia varten asennettiin orkestrointityökalu, joka mahdollistaa palvelinkeskukselle ominaisten toimenpiteiden automatisoinnin. Työkalulla generoitiin palvelimien provisiointiin soveltuva työnkulku (*workflow*), jolla pystyttiin myös luomaan uusi palvelin, ja tälle nimipalvelutietueet, levypinnan allokointi sekä verkkokonfiguraatio.

Lopuksi esitetään kehitysideoita ympäristölle, muun muassa ympäristön päivityksien osalta, IPv6-verkon käyttöönotosta sekä palvelinkonfiguraatioiden automaattisesta hallinnasta. Konfiguraatioiden automaattinen hallinta on infrastruktuurin automaatiosta riippuvainen.

Työ osoittaa, että ohjelmallisesti ohjattava palvelinkeskus voidaan luoda toteutuksessa esitetyillä komponenteilla. Kaikkien komponenttien on kuitenkin tarjottava niiden hallinnan automatisoinnin mahdollistava rajapinta tai jokin muu keino. Tässä työssä esitetty toteutus olisi mahdollista toteuttaa myös muilla laitteisto- ja ohjelmistokomponenteilla.

PREFACE

The method described in this thesis is a completely new way of doing IT operations, and it is very closely related to my work as a Systems Specialist. One of the biggest challenges in the making of this thesis were the arrangements between my work and spare time. The written part was done on my spare time, and the actual implementation presented in this thesis was done during work for my employer, Ambientia Oy. This thesis was written between December 2013 and May 2014.

I would like to thank especially my supervisor, Matias Mäkinen, who suggested this topic for my thesis and gave me priceless feedback and inspiration. Thanks to Professor Jarmo Harju I kept getting good advice during the process. Last but definitely not least I would like to thank my parents and my girlfriend, who have provided endless support, even with my absurd schedules.

In Tampere on May 2014

_____

Ville Törhönen

CONTENTS

LIST OF TERMS AND ABBREVIATIONS

API	Application Programming Interface, a software interface which specifies how a software component can be interacted with each other.
DAS	Directly Attached Storage, which means storage, that has been directly connected to a machine HBAs.
Fibre Channel	FC is a network technology, which allows the use of high-speed networking for storage needs though a HBA.
HBA	Host Bus Adapter is a controller, that connects various network and storage devices to a host system.
HTTP	HyperText Transfer Protocol is a stateless application protocol designed for the World-Wide Web.
Hypervisor	Software, that emulates hardware and which hosts one or more virtual machines.
IaaS	Infrastructure as a Service is a cloud-service model, where the service provider provides virtual machines and other resources as a service.
IDS	Intrusion Detection System is a system used to detect malicous network activity.
IPS	Intrusion Prevention System is a derivative of an IDS, which also prevents the detected activity.
iSCSI	Internet Small Computer System Interface is a SCSI-based network transport protocol on top of TCP.
JSON	JavaScript Object Notation is a data format designed for interchanging attribute-value paired data, generally over the Internet.
NAS	Network Attached Storage is a storage system, which provides file-level access to a storage via a network interface.

NFS	Network File System is a file system protocol, which runs over UDP or TCP protocol, providing file-level access to a storage.
PaaS	Platform as a Service is a cloud-service model, where the service provider provides a computing platform as a service.
RAID	Redundant Array of Independent Disks is a storage technology, which transforms multiple disks into a logical storage element. Benefits include redundancy and performance.
SaaS	Software as a Service is a cloud-service model, where the service provider provides access to applications and databases as a service.
SAN	Storage Area Network is a private network, which is used to provide storage device access.
SAS	Serial Attached SCSI is a serial protocol based on the SCSI protocol, designed for storage devices.
SCSI	Small Computer System Interface is a standard for connecting peripheral devices to a system.
SDN	Software-Defined Networking is a data center architecture, which decouples network control and forwarding functions and enables programmable infrastructure. All networking elements are virtualized.
SDDC	Software-Defined Datacenter is an infrastructure architecture model for a datacenter to achieve virtualization benefits storage, compute and network resources.
SDS	Software-Define Storage is a form of storage virtualization, which offers a unified interface for storage control, enabling a programmable storage infrastructure.
SNMP	Simple Network Management Protocol is a protocol, which enables remote device management and monitoring.

SSH	Secure Shell is a network protocol designed for secure communication between a server and a client.
RU (U)	Rack Unit, U or RU, is a standard of height for rack mounted equipment, which equals 4,445 centimeters.
Virtual machine (VM)	A machine, that runs in a virtualized environment with virtual hardware, within a hypervisor.
VLAN	Virtual local area network is a technique specified by IEEE 802.1Q for isolating multiple broadcast domains over a single layer 2 network.
XML	Extensible Markup Language is a language for defining and representing a document with a structured schema.

1. INTRODUCTION

Over the last 20 years the IT industry has gone through a series of virtualization innovations. In 2006 AMD and Intel released CPU instruction sets which allowed full virtualization, which completely simulated the underlying hardware [1]. This allowed virtualization platforms to run multiple heterogenous operating system instances within a virtualization host, a hypervisor. Since then similar virtualization innovations have been implemented with networking devices and storage systems.

1.1. Motivation

A datacenter, which provides hosted servers or services can be called as a hosting environment. It is a complex environment, which consists of server, networking and storage hardware, efficient air-conditioning and redudant power. Due to the amount of power-intensive hardware, the energy consumption is high. Energy-efficient solutions are more and more popular.

Virtualization has made it possible to provide on-demand services, better scalability, fault tolerancy and high-availability. It has also reduced power consumption due to better server consolidation. During the last few years cloud computing has become a large business in the IT industry. Cloud computing provides scalable infrastructures, automation, and different kinds of service models for different needs. It also reduces costs [2]. According to analytics by International Data Corporation (IDC) cloud computing will transform the core of IT industry within 20 years by a 26 % annual increase of cloud service usage [3].

Traditional hosting technologies are not sufficient to serve cloud-based services. In order to implement automation in service provisioning, all the components in the hosting infrastructure need to be taken into account [4]. Many components are already capable of such automation, if virtualization is being used in the environment. However,

legacy hardware, such as network devices make it challenging to fully implement the provisioning. Also hardware, that doesn't support any kind of automation means that it is a part of the process that has to be done manually.

Infrastructure automation also has an important role in agile software development. Agile software development requires fast feedback, which means software build, deployment and testing has to happen seamlessly. Continuous delivery is a very popular term to describe it, and automated provisioning is a prequisite for such process. This presents challenges for IT operations. A popular term for describing the seamless interaction between IT operations and developers is *DevOps* (Development and Operations). [5]

One example of such automation is the process of provisioning a new server. When a server is deployed it is given CPU and memory resources. These resources are allocated from an virtualization host. After deployment, it has to automatically generate a network configuration to itself and determine which storage it has to use. This process has to be as fast as possible. In terms of continuous delivery, the server has to be able to setup a software environment, which is only possible if infrastructure automation is used.

In order to implement such automation, the whole hosting environment from hardware layer to application layer has to be designed and built to support automation. Virtualization, networking, storage and management are the main aspects of the process. Instead of upgrading a pre-existing datacenter to an SDDC, a new hosting environment has to be designed and implemented. The design has to include distinct and clear building blocks architecture-wise. This makes it possible to build similar environments with the same set of features. However, the architecture can be adapted and re-used with different server hardware.

Software-defined datacenter (*SDDC*) is a term invented by VMware Inc. to represent such architectural design for a datacenter. It includes architectural decisions in computational, networking and storage virtualization [6]. Networking virtualization can be called as software-defined networking (*SDN*) and likewise storage virtualization as software-defined virtualization (*SDS*).

1.2. Objectives

The objective of this thesis is to investigate how a software-defined datacenter can be designed and implemented by using VMware virtualization. Even though the methods described here are specific to a VMware virtualized infrastructure, other virtualization technologies were evaluated and they were proven to provide the same kind of features. Other virtualization environments, similar to VMware, are presented as an option as well.

1.3. Structure of the thesis

Theory of a software-defined datacenter is presented in Chapter 2. Limitations of common hosting technologies compared to an SDDC are also presented. An implementation of an SDDC is presented in Chapter 3, which includes the architectural desicions of the SDDC in question, automation implementation and resource management.

Additional issues and future development is presented in Chapter 4. This chapter presents issues that were not yet implemented. Chapter 5, the final chapter, presents the conclusion.

2. SOFTWARE-DEFINED DATACENTER

In this chapter, the theory of a software-defined datacenter is given. Key components and technologies are introduced and reviewed. First, limitations and challenges of common hosting technologies are discussed. Server virtualization platforms are then introduced and reviewed. An architecture of a software-defined datacenter is then introduced by presenting VMware virtualization technology and its SDDC capable features. NetApp storage system is then introduced, which provides the storage infrastructure to an SDDC. Lastly various networking solutions are presented. Various security standards are also evaluated.

2.1. Overview of the concept

A software-defined datacenter is an architectural model to design a datacenter. It includes "the ability to abstract compute, storage, networking and security resources so that virtualized resources can be deployed and managed in a highly automated fashion" [6]. This enables the usage of various cloud computing models, such as IaaS (Infrastructure as a Service), PaaS (Platform as a Service) or SaaS (Software as a Service). In general, it enables IT efficiency and agility, which cannot be achieved with common and previously used architectures or technologies [7].

SDDC is also closely related to an operational model called ITaaS (IT as a Service), where IT organization is run as a separate business, serving cloud-based services (both internally and externally) via self-service portals [8]. The infrastructure to enable such model can be achieved by a software-defined datacenter.

Virtualization can be achieved with various technologies, which require design choices both hardware and storage wise. These technologies can be divided into two groups: native virtualization, and hosted virtualization.

In native virtualization, the virtualization hypervisor is run as the main operating

system above hardware layer. All virtual machines are run above hypervisor layer. In hosted virtualization, a base operating system is required to run the hypervisor, which thereon runs the virtual machines. This causes extra overhead, as the hypervisor needs to use the virtualization services provided by the underlying operating system.

In an enterprise environment native virtualization has become a de facto. It gives a higher virtualization effiency, availability and also improves security, as an extra layer of software has been removed.

2.2. Challenges in common hosting technologies

Before virtualization or cloud computing, hosting was run by shared or dedicated servers. Server deployment had to be done manually, which was time-consuming. Applications, which needed to be run on a separate server, had to have a dedicated physical server. In a large datacenter this would implicate large costs on server hardware, high demand of physical space and electricity [9]. This also created single point of failures, where a single hardware failure (such as memory failure) could affect all services within a datacenter.

Server virtualization simplifies these problems. Virtualization makes it possible to consolidate servers within servers. With dedicated server hardware low utilization is a problem. More machines can be fit within the space that was earlier used to host single server hardware. This implicates less servers in total, as more servers can be hosted within a single server. Therefore costs are lower on server hardware and repairs. [10]

In addition, virtualization enables various mechanisms, that were previously hard to achieve. High availability, disaster recovery and high scalability are enabled by the virtualization platform. By using shared storage or SANs, virtualization hosts can be used in a cluster, where a single host failure would lead to the relocation of its virtualization hosts without service disruption.

Cloud computing makes it possible to run different kinds of services and service models on top of a virtualized infrastructure. Cloud computing is heavily based on software solutions, which provide different kinds of automation to enable agility and flexibility. This has created large businesses, such as Amazon Elastic Cloud, which provides cloud services to clients ranging from individuals to large enterprises [11]. It has also created businesses, which serve different cloud service models, such as SaaS on

top of Amazon's cloud platform. The platform provides tools and APIs to setup services and manage them.

Virtualization itself has its disadvantages, as it only applies to a part of the infrastructure. Many tasks are much faster than with regular hardware, but the basic infrastructure, which comprises of switches, routers and storage systems are still the same which was being used in a non-virtualized infrastructure. This causes manual processing problems and overhead, which is caused by unautomated and non-virtualized hardware. [12]

In a software-defined datacenter all components (or layers) of the datacenter are virtualized. In addition to virtual machine deployment, network infrastructure and storage systems are controlled by software.

2.3. Virtualization platforms

Server virtualization can be achieved by many different technologies. Therefore, an SDDC can be built by using other virtualization platforms than VMware's vSphere. Virtualization platforms, such as Citrix's XenServer or Microsoft's Hyper-V, include similar functionalities crucial in an SDDC. For example, support for orchestration tools and features such as machine live migration are required. The software components in the infrastructure are almost identical across these environments.

In this thesis, an SDDC is presented and implemented by using VMware technology. This technology was chosen, at it was already heavily used in a previous environment for years before new datacenter design was started. Another important part of this decision was compatibility with the previous environment. All virtual machines from the previous environment should be compatible with the new environment as-is. This minimized extra work, reduced costs and minimized risks.

2.4. Architecture and components

The virtualization platform consists of datacenters, clusters and hosts. In VMware, the platform is called vSphere. In a physical sense, a datacenter is a facility, which holds all server hardware, storage and networking equipment. An SDDC consists of one or more virtualized clusters, which are formed by virtualization hosts. A cluster is a transparent

group of machines, which implements high-availability and scalability. It can also be considers as a resource pool.

As defined above, a cluster can be formed of multiple virtualization hosts. When in cluster, these hosts share information about virtual machines in the cluster. In case of a virtual machine failure or complete virtualization host failure, all virtual machines are relocated to a functional virtualization host.

A virtualized VMware infrastructure, or vSphere, consists of the following key components:

- Bare-metal virtualization hosts, which are called hypervisors. Only minimal local storage is needed, as hypervisor is run in-memory and all virtual machines are run on and from a shared storage system. These are called ESXi hosts in VMware.
- Storage system, where all data resides at. All virtualization hosts within a cluster must have access to the same data devices. Data devices are called datastores, which are attached to ESXi hosts.
- Management node, or multiple nodes, which control and configure the infrastructure. In VMware, this is called vCenter.

Virtualization platform itself is not dependent on the virtualization host hardware. VMware ESXi runs on a range of different vendors and CPU architectures. Compatibility between the hypervisor software and server hardware can be determined. However, to prevent compatibility problems between hosts, it is advised to use homogenous server hardware within a cluster.

2.5. Virtualization platform management

VMware vCenter Server is used as a central management node for VMware virtual infrastructure. vCenter can be deployed to its own dedicated hardware, or it can be deployed as a virtual machine to the virtual infrastructure it manages. It can control multiple datacenters, hosts and clusters. In addition to its management capabilities, it provides datacenter-wide visibility and monitoring. [13]

2.5.1. Virtual machine provisioning

Virtual machines can be provisioned via vCenter. As vCenter is connected to all virtualization hosts in the infrastructure, virtual machines can be assigned to a specific host in a specific cluster. A virtual machine must be located to a datastore, which is usually a specific data device from a storage system. A directory will be created for the virtual machine on the target datastore, which will hold all configuration files and disks the machine will use. It is possible to add multiple disks during provisioning, and also determine size for each disks. Any peripherals, such as network interface cards, can be attached to the virtual machine as well. All configuration is written to a configuration file within the directory, which defines virtual machine settings such as CPU and memory allocation.

Usually virtual machines are not provisioned without a pre-installed operating system. Otherwise a manual operating system installation would be required. Even though the installation could be automated with various techniques, a much faster way is to use virtual machine templates as a baseline for new virtual machines. These templates include disks with pre-installed operating systems and pre-configured peripherals. Instead of creating a new virtual machine from the beginning, it is possible to deploy a new virtual machine from a template. When the virtual machine is started, it only requires operating system specific configuration.

Third way of provisioning a virtual machine is to use cloning. It is a similar operation as deploying from a template, but any pre-existing virtual machine can be used as a template. This is especially useful in cases where a similar virtual machine needs to be created from an already-configured virtual machine with identical hardware and operating system configuration.

2.5.2. Virtual machine migration

Virtual machines can be migrated from one host to another by using a feature called VMware vMotion. This is useful when resources need to be spread across a cluster or a datacenter equally. This migration can be done on-the-fly while the virtual machine is powered on if the migration is done within a cluster. As both the source and the target virtualization hosts have access to the same storage, they can change the host of

a virtual machine. This is called live migration. The migration is started by copying virtual machine's physical memory to the target host. This is done in three phases [14]:

1. Guest trace phase. This is when the migration is initialized by setting markers to the guest memory pages. This is necessary, as the virtual machine is powered on, and its memory pages are continuously changing.

2. Pre-copy phase. Virtual machine memory pages are copied in iterations, first by copying the whole memory map and then only the pages that were changed since the last iteration.

3. Switch-over phase. The virtual machine is quiesced for a while in order to copy changed memory pages since the last iteration. After copying, the source virtualization host unregisters the virtual machine and the machine is registered on the target host.

State of the machine's CPU, network and other virtual devices are also copied. All active network connections are also preserved on the migration. After all copying is done, the virtual machine is resumed as-is on the target virtualization host without disruption.

If the migration has to be done from cluster to another, or from datacenter to another, the virtual machine has to be shut down when running vSphere version older than 5.1. As shared storage is not preserved across clusters or datacenters, virtual machine data has to be copied. Comparing to live migration this is a more straight-forward operation, as only the data is copied. From vSphere 5.1, VMware introduced a technology called Storage vMotion, which enables virtual machine live-migration from cluster to another, or even from datacenter to another. It also enables live-migration from a datastore to another without changing the virtualization host. This is especially useful when a virtual machine must be migrated from disk tier to another, which means migrating from a storage backed by slower disks to a storage backed by faster disks. This makes it possible to run virtual machines on different service levels.

2.5.3. Resource pooling

Resource pools can be used to manage which virtual machines can use resources, such as memory and CPU. This is called abstract resource partitioning. Resource pools are

created within a cluster or a single virtualization host. With resource pools it is possible to create groups of virtual machines, where a set of resources are given and it is shared among the virtual machines in a best effort manner. This enables the usage of different kinds of service levels. It is possible to give dedicated resources to a virtual machine, when another machines hosted on the same virtualization host are unable to use those resources.

2.5.4. High availability

Cluster high availability can be achieved with VMware High-Availability module. It is implemented in the vSphere virtualization platform, hypervisor-level high availability agents and even in the virtual machine operating systems [15]. Implementation is based on continuous heartbeat monitoring. In case of a virtualization host failure, other virtualization hosts notice that certain virtual machines need to be relocated. This is called a failover operation. If an operating system stops responding to the HA agent heartbeat, the virtual machine can be restarted by the HA agent.

These functionalities are heavily based on shared storage, as all virtualization hosts can access the same virtual machines exclusively. In case of a failover, the HA agent ensures that sufficient resources are available in a resource pool at all times. This is done by doing continuous resource monitoring within the cluster.

2.5.5. Scalability

Clusters can be scaled vertically by adding more virtualization hosts to the cluster. VMware HA has a maximum of 32 hosts per cluster [16]. If VMware HA is used with VMware Distributed Resource Scheduler (DRS), all resources are spread across the cluster optimally. If needed, virtual machines are migrated from one host to another to balance resource usage between the hosts in the cluster.

Scalability is an important factor during cluster maintenance operations. Typically a virtualization host has to be restarted, for example when server hardware is repaired or upgraded, or when the hypervisor software is upgraded or patched. All virtual machines that are located on the host have to be migrated to another host. This is not possible if the cluster capacity is overloaded. It is necessary to keep capacity balanced, so that at least a

single virtualization host can be cleared out of virtual machines. [17]

2.5.6. Orchestration tools and automation

Automation is the key aspect of SDDC. On-demand services such as cloud computing and fast service provisioning can be achieved by implementing orchestration. Kaltz, C. describes orchestration as "an executable business process that can interact with both internal and external web services" [18]. The actual orchestration is executed by the orchestration engine. Orchestration engine executes workflows, that describe the actual business process. An orchestration workflow can utilize APIs from compute, storage and networking layers. It has to be customizable and extendable, as more and more systems have to be integrated into the workflow automation.

Orchestration can be implemented by using VMware automation products or the vCenter application programming interface (API). The API is accessible from various software development kits (SDK), and it is also utilized by the automation products. Administrators can run pre-defined workflows in VMware vCenter management application without any additional configuration. An example of a pre-defined workflow is virtual machine cloning, where a pre-existing virtual machine can be replicated to a new virtual machine. Custom workflows can be implemented by programming custom software or scripts, which utilize the vCenter API. However, to implement flexible and customizable automation, VMware vCenter Orchestrator (vCO) can be used to create custom workflows.

VMware vCenter Orchestrator is an independent, Java-based web application, which is connected to a VMware vCenter Server. It utilizes the vCenter API to control the virtual infrastructure presented by the vCenter. Orchestrator can be connected to additional vCenter Servers running in different environments. An example architecture is shown in Figure 2.1, where a single Orchestrator instance controls a single vCenter. When vCenter receives an API call from Orchestrator, it controls the target virtualization host or hosts over a dedicated management network. Orchestrator can also manage cluster configuration and initialize operations within the virtual infrastructure, such as virtual machine live migration.

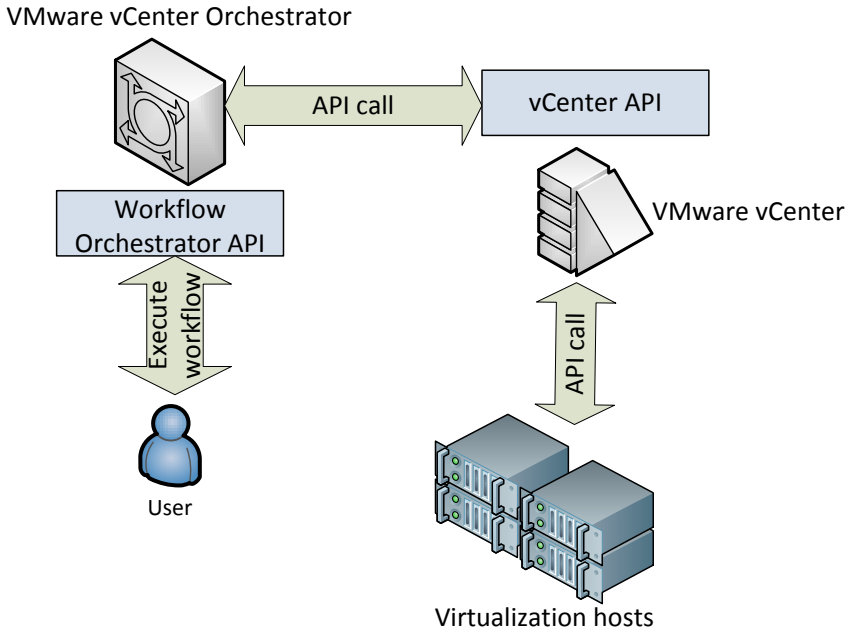

Figure 2.1: Architecture of a VMware virtual environment controlled by VMware vCenter Orchestrator. As an example, API calls related to workflow execution are presented.

Orchestrator provides the ability to define custom workflows and a Javascript scripting engine. All configuration is done via a Java-based Orchestrator desktop client by connecting to the Orchestrator. Workflows are built and designed with a graphical interface. Input and output parameters can be defined. An Orchestrator workflow consists of actions, which are run step-by-step. Actions range from basic iteration statements and pre-defined API calls to custom actions, which are written in Javascript. This allows to create complex actions, that can utilize any API call presented by the vCenter. Error-handling can be implemented with exceptions or by reading workflow action return codes. [19]

Workflows can be executed with the desktop client. In order to implement software-controlled automation, the Orchestrator provides a workflow-specific API. All workflows have a unique identifier. This allows remote workflow execution by sending a HTTP POST request to the Orchestrator by targeting the API call with a workflow identifier. Workflow input can be sent in either XML or JSON format.

2.6. Data storage systems

Data storage and management is a key aspect in an enterprise datacenter. The most common storage system vendors are Dell, EMC, Hitachi and NetApp [20]. This thesis

is based on NetApp specific technologies, which can be applied to other vendors as well but with different terminology. In NetApp, the storage controller is called a filer, which runs Data ONTAP operating system. The storage API is called Data ONTAP API and it is accessible over HTTP protocol.

The most simple approach to implement storage is to use local storage within the virtualization hosts, such as local SAS or SSD disks, directly though a HBA. Local storage is a form of directly-attached storage (DAS). This approach has its risks, as data resides on one virtualization host, and if the host has a hardware failure, that data is lost. Some improvements can be achieved by using storage resiliency, such as RAID. In addition to this, local storage is not as scalable as other options and it is challenging to backup.

Instead of using relatively high-risk local storage, the most common way for achieving storage high-availability is to use a storage system. A storage system has a separate storage controller and one or more disk arrays attached to it. A disk array consists of homogenous disks, which can be either SATA, SAS or SSD disks. Physically disks are located in a disk shelf, which is connected to the controller itself. Disk shelves can be attached to the controller with various protocols, such as FibreChannel and SAS. A storage system can be called as a SAN device (Storage Area Network) or NAS (Network-access storage), depending on the use case. [21]

All physical storage elements of a storage system are consolidated into logical storage layer. This can also be called as storage virtualization [22]. Storage allocation and provisioning can be done with various protocols, such as FibreChannel over Ethernet, NFS or iSCSI, from the storage system to the hosts. Storage system can provide either block-level or filesystem-level access to the storage volumes. A host can be, for example, a virtualization host or a virtual machine within the host. In NetApp, all physical disks are part of a RAID group. RAID groups form an aggregate, to which volumes can be provisioned to. Volumes can be enlarged or shrunk online, by utilizing space from the aggregate it resides on. These volumes can be exported as-is to the host. It it also possible to create separate directories within volume, or qtrees in NetApp terminology, which can be assigned a separate quota. These qtrees can be exported separately with their own access lists. A volume can also include iSCSI LUNs (Logical Unit Number).

In an SDDC, the virtualization platform heavily relies on the storage infrastructure

and its capabilities. As with the virtualization layer, the storage layer must have an API. This enables storage provisioning, which can be utilized when provisioning a virtual machine. The API gives a unified representation of the storage layer, which notably eases automation and software development.

2.6.1. Clustering and fault tolerancy

In order to prevent single-point-of-failures, data storage systems can be clustered by clustering storage contollers. In NetApp Data ONTAP, the storage controllers can form a cluster in Cluster-Mode or 7-Mode. If the storage system only consists of two storage controllers, it is a 7-Mode Cluster and therefore a HA pair (High Availability) [23]. In a clustered environment, the storage controllers are called as "nodes". If there are more than two storage controllers, then the cluster is formed by HA pairs. However, only the nodes of a HA pair can takeover each other's storage [24]. Storage controllers are connected to all disk shelves within the HA pair, but also to the other contoller in the HA pair. All nodes within the cluster are connected to each other with cluster interconnect. A 7-Mode cluster is shown in Figure 2.2.

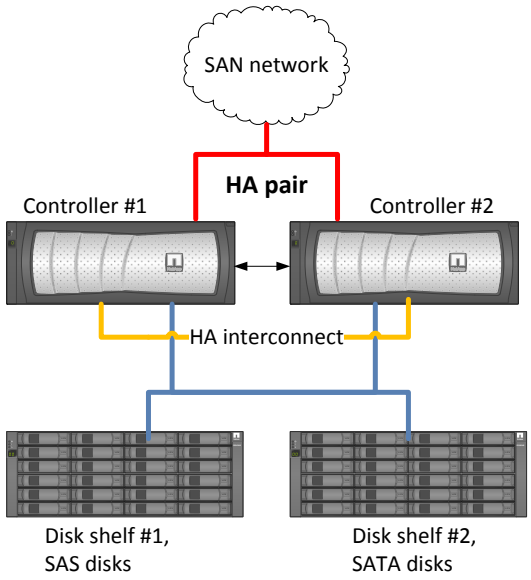

Figure 2.2: A HA pair (Data ONTAP 7-mode) of two NetApp FAS-2240 controllers. Both controllers are connected to both disk shelves, and they are connected to each other with an interconnect link.

Clustering has also other advantages than just high-availabilty, as it also enables

scaling-out storage. When a non-clustered storage system, or a single controller, is at its maximum capacity (both CPU and memory-wise), there are two options how to gain more capacity and performance. The storage controller can be replaced by a more powerful controller, or a new controller can be run in parallel with the old one [23]. Both of these can be called as scaling up, which means bringing new separate hardware to the environment. This solution does not scale easily and it can cause downtime. It is also hard to manage two separate storage systems. In a cluster, this can be achieved by joining additional storage controllers to the cluster. This does not cause downtime. It is also possible to add homogenous HA pairs to the cluster, meaning that new hardware can be added to the cluster, forming a new HA pair. An example of a four node cluster is shown in Figure 2.3.

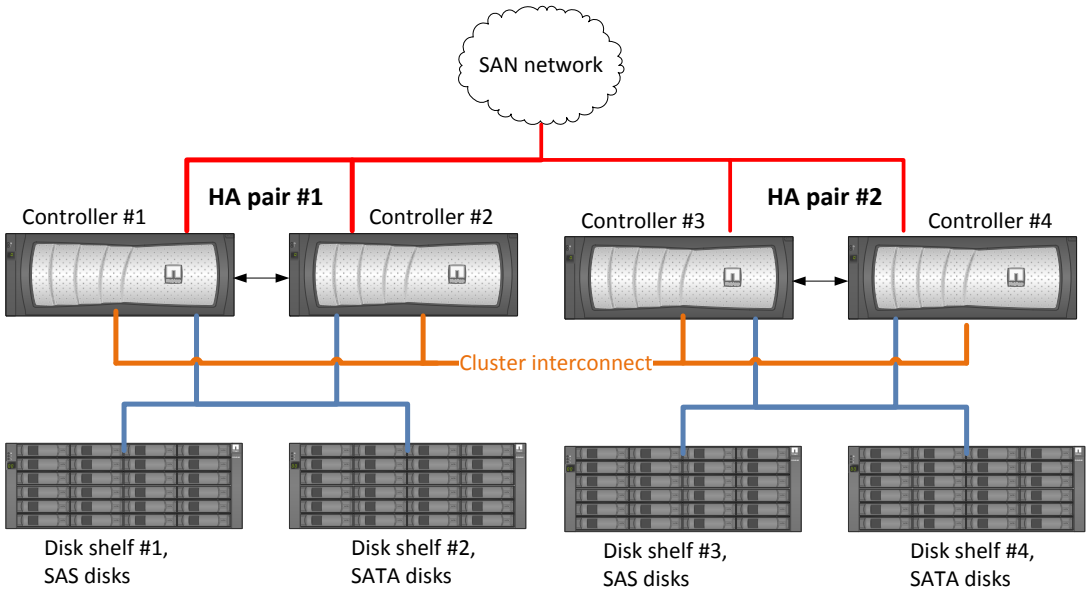

Figure 2.3: A four node Data ONTAP Cluster-mode storage system, which consists of two HA pairs and four disk shelves. As with the two-node HA pair, this has an interconnect network as well between all cluster nodes.

Fault-tolerancy and disk resiliency within the disk shelf is implemented by double-parity RAID technology [25]. This technology is NetApp's implementation of a RAID 4 group with additional parity. In a traditional RAID 4 group, a single disk is used as a dedicated parity disk. Data is stored in horizontal stripes, and its parity is calculated to the parity disk. In RAID-DP, an additional parity disk is added, which holds diagonal parity calculations from diagonal stripes. Compared to a single-parity RAID group, double-

parity offers protection levels for two failed disks within the RAID group.

Capacity requirements and modern disk sizes are increasing. Fewer and larger disks are generally utilized in a storage system, rather than utilizing many smaller disks. In case of a disk failure, the disk has to be replaced. If the disk did not fail, but a read error occured, the erroneous bit can be recalculated immediately. However, when a drive is replaced, the RAID group has to be rebuilt. RAID group rebuild is done by recreating data to the new disks from unharmed disks and parity disks. During this phase the RAID group is in a vulnerable state. If another disk failure occurs, data is lost. Rebuilding the RAID group is a time-consuming process and it can take days. The larger the disk, the more it takes to rebuild the RAID group. With RAID-DP, the second drive failure does not cause data loss because of the diagonal parity calculations.

2.6.2. Flash-based disk caching

In a virtualized environment, much of the performance comes from the underlying storage. Capacity can be increased easily, as disk drives are getting bigger and bigger in capacity. When capacity isn't a problem, sooner or later disk performance becomes an issue. Much of the disk performance comes from the disk latency. If the disk has a small latency, then its throughput is bigger. One solution is to use a lot of small but fast disks. However, this wastes physical space and it is costly electricity-wise. Replacing a disk-shelf with faster drives and the same data is costly as well, and the operation requires downtime. SSD drives are much faster than magnetic drives and their capacity per price is still much higher than on magnetic drives. Kang et al. suggest [26], that flash-based storage, such as SSDs, are "economically more sensible to supplement disk drives rather than replace them".

In order to increase disk perfomance, a flash-cache can be applied to the storage system. Storage system utilizes flash-cache as a read cache, where the most wanted data blocks are kept in the fast, flash-based memory instead of on the primary storage backend run on magnetic disks. When a read request is given to the storage contoller, the storage controller tries to read the data first from the flash-cache. If it is not there, the data is read from the storage backend. In a cluster mode, all storage controllers must have a own dedicated flash-cache. Flash-cache can vary in size, starting from 512 GB. According to

NetApp, flash-cache read hits reduce latency by a factor of 10 [27].

2.6.3. Storage automation

In order to implement an SDDC, automation must be available to be used in the storage layer. A use case for such situation would be a virtual machine deployment, which will store its data on an NFS export. This means that the storage system must be commanded to create either a volume or a qtree, determine its size and create an export for the virtual machine. The most efficient way to achieve this is to integrate it with the orchestration tool, which in this case is VMware vCenter Orchestrator. During virtual machine deployment, the orchestration tool will then utilize both virtual infrastructure API and the storage system API.

NetApp Data ONTAP provides an XML-RPC API, which provides all the same functionalities that are available in the ONTAP command-line interface. This API can be used via a SDK (Software Development Kit) provided by NetApp, called NetApp Manageability SDK (NMSDK) [28]. This provides an easy-to-use API libraries for different programming languages, such as C, Java, Python and Ruby. SDK is free for NetApp customers and partners. Before the API can be utilized, it must be enabled on the storage controllers.

2.7. Networking technologies

The third important part of an SDDC is the network equipment. A software-defined datacenter utilizes the same components and protocols used in hosting environments generally. For example, VLAN networks are used in order to isolate networks from each other. In order to provide fault-tolerancy in network links, protocols such as LACP can be used to implement link-aggregation [29].

Traditionally network equipment has been managed device-by-device, or by utilizing vendor-specific management applications. Many of these applications are beneficial only when operating in a large environment. Some management applications, such as Hewlett-Packard Intelligent Management Center, support devices from other vendors as well, and provide limited support for managing them [30].

Generally management applications rely solely on remote console access to the

network equipment. All functionalities are implemented by running commands to the network device via a remote connection, such as an SSH session. This causes various problems, such as device-specific implementations, vendor lock-in and unwanted heterogenous configuration in a homogenous environment. In order to prevent this, a vendor-independent management method must be utilized.

Ideally, an SDDC utilizes SDN (Software Defined Networking), which is a type of architecture concept for controlling and implementing networking in a virtualized manner. This architecture is described in Figure 2.4. In SDN, the physical network equipment is controlled by a controller, which is then controlled by software. The infrastructure can be grouped in the following layers: application layer, control layer and infrastructure layer. This architecture enables centralized management, even in a multi-vendor environment. More importantly, it also enables networking and virtualization platform integrations in the orchestration software.

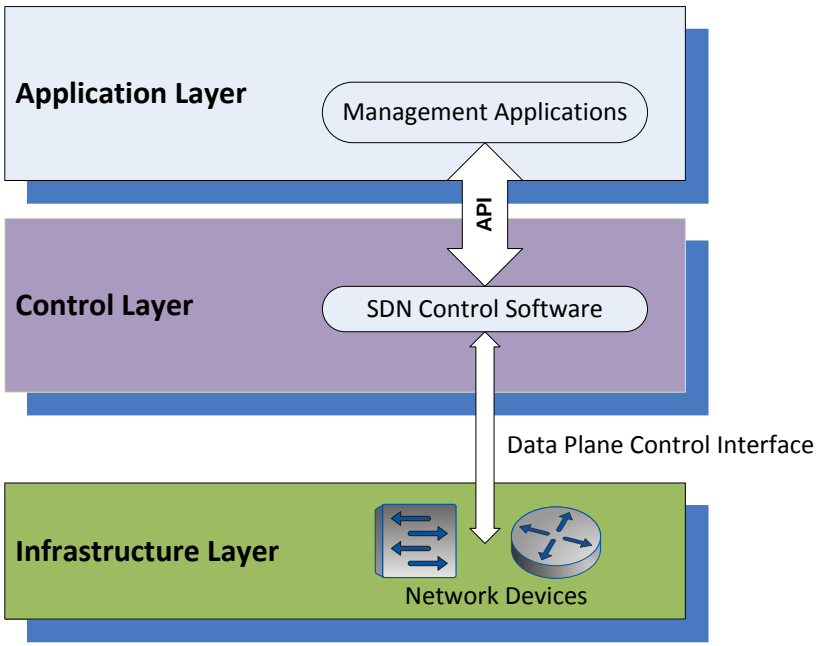

Figure 2.4: Architecture layout of Software-Defined Network consists of an application layer, a control layer and an infrastructure layer.

As an architecture the software-defined networking has been specified by Open Networking Foundation [31]. The basis for such specification is the standardized OpenFlow networking protocol, which defines a unified data plane interface for networking equipment. All devices which implement the OpenFlow protocol can be controlled with the same protocol. Ideally, this means that the actual flow processing, for

example routing, is done on the control plane. A controller is used instead of the device. However, network equipment, that does not support OpenFlow, can still be controlled by means of software. Proprietary protocols have been made as an option to OpenFlow, such as Cisco's Application Centric Infrastructure (ACI) [32]. This thesis only focuses on the architectural model of SDN.

Another network architecture concept for an SDDC is Network Functions Virtualization (NFV). In NFV, all network functions are virtualized and run within the virtual infrastructure it manages. Instead of using physical network equipment, the functionalities are implemented in a virtual appliance. With this network infrastructure the virtualization layer benefits, such as high availability and scalability, can be applied to network equipment. NFV has some similarities with SDN, but in NFV network implementation is not separated into different layers, but completely virtualized. The architecture itself is not dependent of SDN, and both of them can be considered complementary to each other. [33]

2.7.1. Switch virtualization

Traditionally, a datacenter has a core network, which consists of multiple physical switches. The core network consists of multiple virtual networks, which are routed within the core switches. These networks are usually hosting or SAN networks. Layer 2 network protocol, STP (Spanning Tree Protocol) or its derivatives RSTP (Rapid Spanning Tree Protocol) or MSTP (Multiple Spanning Tree Protocol) can be used to enable link redundancy between switches and other network equipment, such as routers or firewalls. If a link fails, secondary backup paths are available. These protocols are primarily used to detect and protect bridge loops between switches and to provide high availability.

The original STP protocol is standardized by IEEE's 802.1D. The protocol is well supported with different network devices. However, STP has its disadvantages. In various cases, the spanning-tree algorithm (STA) can perform unwanted port blocking or forwarding. Such cases are usually configuration issues with the STP port configuration. Configuring and maintaining STP in a large network can be time-consuming and error-prone. If one of its derivatives is used, the configuration is even more complex with some performance benefits. If additional switches are brought to the environment, the

configuration has to be done very carefully.

Hewlett-Packard introduces a solution to this problem with IRF (Intelligent Resilient Framework). It is a properietary technology, which allows switch virtualization over multiple (up to 9) physical switches. With IRF, the switches form a cluster, where one switch is the control node of all the other switches. Switch cluster is called as an IRF fabric in HP terminology. The master node is determined by cluster election, whereas the others work as subordinates. [34] This is similar to the architectural model described by the SDN, where the physical plane is separated from the control plane. Configuring the switch is therefore easy, as instead of configuring the switches separately, all configuration is done from the master node of the switch cluster. All switches are seen as one, in terms of configuration. Even though the architectural model is similar as with SDN, it is unclear whether or not IRF utilizes OpenFlow in its virtualization implementation.

The IRF fabric can be formed by either using a daisy chain topology or a ring topology. Ring topology provides more reliability, providing fault tolerancy to a single link failure. A ring topology, consisting of four switches, is presented in Figure 2.5. Each switch is connected to another switch by using either a single link or by multiple links, aggregated into a one logical link on both ends.

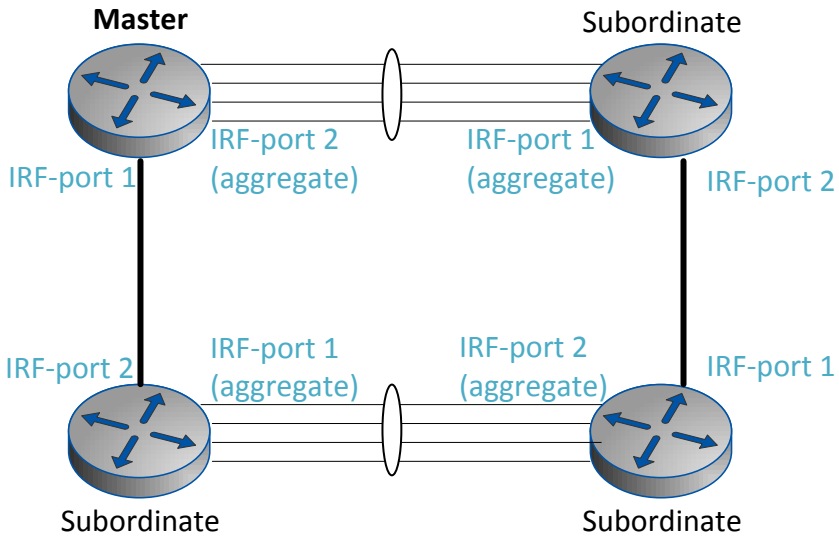

Figure 2.5: An IRF fabric in a ring topology. All switches have two neighbours, which are connected either by a single link or a link aggregate, consisting of multiple physical links.

2.7.2. Virtual LAN (VLAN)

A network can be partitioned on the data link layer with a technology called Virtual LAN, or VLAN. This technique has been standardized by IEEE's 802.1Q. Partitioning is done by creating multiple isolated broadcast domains, which are identified by a VLAN tag ranging from values 1-4096. This isolation provides security, as all traffic targeted from a VLAN to another must be routed via a router, and access control can be applied that way. It also enables scalability and network management, as a VLAN can range over multiple switches. [35]

VLAN enables the use of interconnect links, which can be called as trunks. A single physical network link can be used as an aggregate for multiple different VLAN tags. Tagging can be done either on the switch itself or the end device, if it supports VLAN tagging. A switch port can be used as an access port, when the packets coming from the end device gets tagged only on the switch.

In a virtual environment the virtual infrastructure must support VLAN tags, as virtual machines are usually assigned to different VLANs. An example of such situation is where a virtual machine is assigned to a customer-specific VLAN, which cannot be accessed from any other VLAN. This means that in addition to VLAN tags of the switch ports the virtualization hosts are connected to, the virtualization host itself must work as a switch. In a VMware environment, this is called as a virtual switch, or vSwitch. A virtual switch emulates a traditional switch by relaying packets according to their VLAN tags. A virtual switch is a virtualization host specific component, which is configured by the actual physical network adapters, virtual port groups and virtual ports [36]. A port group determines the VLAN tag of a virtual port.

New VLANs can be propagated in an SDDC by orchestration tools, in this case VMware vCenter Orchestrator. Orchestrator can create new port groups to all necessary virtualization hosts by re-configuring their virtual switches. Depending on the environment, the VLANs can be propagated to the core network either by OpenFlow management software or by custom-tailored software that connects to the master node, creates the VLAN and tags all virtualization hosts' switch ports.

2.7.3. Virtual private network (VPN)

A virtual private network (VPN) is a technique for extending private networks across the Internet. Traffic between these networks are routed via the Internet, securely encrypted, instead of routing the traffic as-is over the Internet [37]. The actual traffic is encapsulated within the encrypted packets. This can also be called as a VPN tunnel. The most widely used protocol to implement such VPN connections is IPsec, which provides various other protocols for creating a specific kind of VPN. IPsec operates at the Internet Layer in the OSI model, below Transport Layer, which makes it possible to tunnel various kinds of transport layer protocols over an IPsec tunnel. VPN connections are primarily used as a site-to-site connection between the hosting provider and the customer. Such setup enables secure connections between hosting and customer networks for specific hosts and networks. Therefore application level encryption is not necessarily needed.

VPN connections are primarily terminated to a dedicated VPN device, which is located in the edge of the network. Packets are decapsulated on the edge network, and then routed through the virtual infrastructure onto the virtual machine. Another option is to create customer-specific VLANs, and route the decapsulated traffic within a dedicated VLAN through the network onto the virtual machine. Both options can be automated. First option requires only device-specific VPN tunnel configuration in the edge VPN device. The second option, in addition to the VPN device configuration, requires VLAN configuration in the core network and the virtual infrastructure. The latter one can be automated, as presented earlier. Device-specific VPN tunnel configuration can be automated with custom-tailored software.

2.7.4. Firewalls and security

A firewall is a device or a software that defines what kind of traffic is allowed to pass through, either in or out, by inspecting packets. A firewall has a set of rules, which it utilizes to filter network packets. A firewall can also manipulate network traffic, for example by utilizing Network Address Translation (NAT) to modify datagram packet headers. Typically a firewall is located at the edge network to protect networks behind it from outside connections. It is also possible for a single device to perform as a firewall and a VPN device. This is very beneficial as these devices can be usually clustered to

provide a HA pair of edge network equipment.

In addition to firewall, the edge network equipment can be used to detect and prevent malicious network activity. This is called an Intrustion Detection System (IDS) or its derivative Intrusion Prevention System (IPS), which prevents the detected activity. Services hosted with unrestricted access, such as public web services gain security concerns and threats depending on the organization. These threats can be detected with an IDS/IPS system by using various methodologies, including signature-based, anomaly-based detection or by stateful protocol analysis. An example of such malicious activity is a Denial-of-Service (DoS) attack, where attacker sends targeted network packets to congest a network link. A distributed version of this attack is called DDoS, where many machines are utilized to send these targeted network packets. An attack like this causes various problems in other hosted services as well. It can also spread to the core network, which is protected by the edge network equipment. Mitigating these attacks can be done with an IDS/IPS system. [38]

A core network is a network, where the actual virtualization environment is run. It includes storage and virtualization hosts, virtual machines and other devices. Core network is connected to the edge network by a transport network, which determines which networks are allowed to be routed from the core network to the edge network. Typically a core network has a router, which can also be the same device as the core network switch. If the firewall is located only on the edge network, traffic within the core network is unrestricted. For example, without traffic filtering in the core network all inter-VLAN connections are allowed. In order to prevent this, the core network switch must also work as a firewall. Even if the core switch does work as a firewall, all intra-VLAN connections remain unrestricted. Machines located within the same network broadcast domain can reach each other without any restrictions by definition.

The aforementioned problem becomes even more problematic in a virtualized environment. Virtual machines are connected to a virtual switch in the virtualization host. Connections within the same broadcast domain or within the same virtualization host cannot be restricted in the core network. VMware provides a solution to this by VMware Distributed Switch, which is an advanced version of the standard vSwitch and which, among other features, enables a firewall in the vSwitch [39]. This makes it possible

to define virtual machine specific firewall rules, even restricting access within the same broadcast domain. The switch also has an API, and it is possible to automate firewall rule management to orchestration tools. However, Distributed Switch requires a special license and it isn't handled in this thesis. An alternative to this is to utilize local operating system level firewalls. Firewall rule propagation can then be automated with configuration automation tools.

2.8. Security standards for data centers

Different kinds of standards have been developed to classify and certify data centers. These standards include areas of information security (ISO 27001:2005 [40]), quality management (ISO 9001:2008 [41]) and environmental management (ISO 14001 [42]). The standards set regulations and requirements, for example, on risk and threat management, physical security, staff competence and also promote environmental friendly decisions. Many of these require regular auditing to ensure the standards are still complied with.

If the datacenter handles payment traffic or stores payment data, it must comply with Payment Card Industry Data Security Standard (PCI DSS). This standard sets requirements on the network infrastructure, data protection methods such as encryption and hashing, firewalls and other access control components. One important issue is network isolation, which means that the payment data must be routed through a dedicated VLAN. Access restrictions to this VLAN must be applied. As with other previously mentioned standards, regular auditing is required. [43] Failure to comply with the standards sets risks to the whole business, and in addition to loss of trust, it can cause financial loss and even legal issues [44].

3. IMPLEMENTATION

This chapter presents an implementation of a software defined datacenter. First, the system architecture is presented. All components, that have been selected to be used, are also presented.

3.1. Architecture

The architecture of an SDDC is the key factor for enabling all the SDDC specific functionalities specified in Chapter 2. In this thesis, the SDDC implementation is a private cloud, not a public or a hybrid cloud. The SDDC presented here is designed for in-house use, enabling in-house IaaS and PaaS functionalities. However, the SDDC here is designed in a way, that could be extended to a hybrid cloud model.

3.1.1. Selected components

In this chapter the basic building blocks of a software-defined datacenter are presented. The hardware described can be easily extended and multiplied. It is also possible to generate another similar SDDC with the components described here. For compute resources the SDDC was chosen to be implemented with Hewlett-Packard's BladeSystem hardware. This solution provided high-availability features and it was compact in size. A c7000 Blade Enclosure was chosen, holding up to 16 half-sized blade servers. The blade enclosure itself is 10 rack units (RU) high. Blade servers were chosen to be HP ProLiant DL380p Gen8, with a maximum of 256 GB of RAM and low-power Intel Xeon E5-2600 v2 processors. Energy efficiency and performance were the key reasons to this decision. With 16 blade servers and with the high-availability the virtual infrastructure provides, a single hardware failure does not affect to the overall reachability. It also enables the use of planned server maintenance, as a single host can be cleared out of virtual machines by utilizing virtual machine migration.

Core network was chosen to be implemented with four Hewlett-Packard 6125XLG Ethernet Blade Switches. These switches were located to the backside of the blade enclosure, each providing 20 internal 10 Gigabit Ethernet ports to the blade servers via blade enclosure back plane, four external 40 Gigabit Ethernet ports and 16 external 10 Gigabit Ethernet ports. These switches were chosen as they provided a scalable and resilient core network with the IRF technology presented in Chapter 2. One 40 Gigabit Ethernet port per switch was reserved for the IRF fabric, in addition to 4 internal 10 Gigabit Ethernet Ports per switch.

The storage system was selected to be a NetApp storage system. Two FAS 2240 storage contollers and two disk shelves were chosen to run as a 7-Mode HA pair. The 24-slot disk shelves were chosen, as they could be easily extended if more capacity was needed. The storage controllers had an internal disk shelf, therefore only a single external disk shelf was needed. No flash-cache drives were selected, but it was kept as an option for future development.

The virtualization infrastructure was selected as VMware vSphere version 5.5. The most restrictions to the SDDC design were caused by licensing issues. For example, the VMware Distributed Switch could not be used. This decision was made due to high licensing costs. Other VMware orchestration tools than the vCenter Orchestrator, such as VMware vCloud Director, could not be selected for use either. VMware management nodes, such as the vCenter Server and the Orchestrator were chosen to be run as virtual machines inside the virtual infrastructure. Both virtual machines were available as a pre-configured virtual appliance from VMware.

Network security was decided to be implemented with Cisco ASA 5525-X devices, which can work both as a firewall and a VPN device. These devices were located to the edge network. Two devices were chosen to be run as a HA pair, where in case of a hardware failure, the stand-by node would take over the connections. It is also possible to extend the HA pair to a cluster by adding mode devices to the infrastructure. This enables scalability on the edge network and enables overprovisoning, where the actual link utilization is always much lower than the maximum throughput. The device also supports IPS, which can be applied to protect single hosts or a specific VLAN.

3.1.2. Logical layout

A simplified network topology was designed to have a clear edge network, core network and a storage network. This was done to simplify network structure between internal and external traffic. The network topology is shown in Figure 3.1.

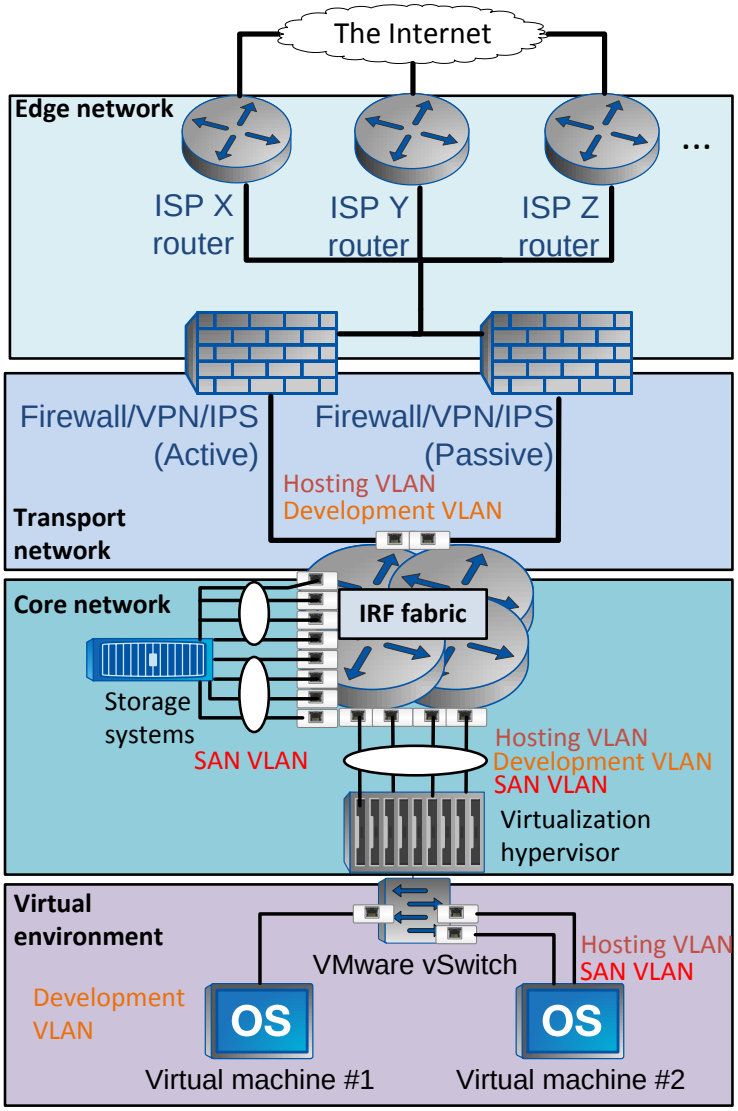

Figure 3.1: Network topology, including routers, firewall devices, IRF switch fabric, storage system and a single virtualization host. As an example, two virtual machines are shown. Both machines are connected to a virtual switch.

Packets are routed from the edge network via firewall, through a transport network, into the core network, where all actual virtualization hosts and storage systems reside at. The core network was implemented as an OSI layer 3 network. In a core network, the traffic is not routed to the edge when the traffic is routed from one network to another within the core. In this case, the IRF fabric within the core was used to maximize the performance,

consisting of four redundant 40 Gbit/s links. The implementation is identical to the layout shown in Figure 2.5.

Virtualization hosts are connected to the switch by four redundant physical links to four switches. These four links are grouped into a link-aggregate. A load-balacing algorithm is used both at the hypervisor and the core switch to balance traffic between the network links and to provide fault-tolerancy. Each storage controller is connected to the switch with four redundant 10 Gbit/s links, also forming a similar link aggregate per storage contoller.

In Figure 3.1 a single virtualization host is shown. It hosts two virtual machines, one of which has two network interfaces. This virtual machine is connected to two networks: SAN and Hosting. These VLANs are tagged on the physical switch ports the hypervisor is connected to. However, only Hosting and Development networks are routed outside the core network onto the transport network, and eventually to the Internet. If virtual machine #1 sends a packet from Development network to virtual machine #2 in Hosting network, the packet is routed via the core network. A firewall can be used to deny such traffic in the core switch. Only virtual machine #2 can access the storage system as it is connected to the SAN network. If virtual machine #1 tries to access the SAN network, the packets are not routed to the network.

Within the core network, the IRF fabric master node computes all the routing. Inter-VLAN traffic is routed through the layer 3 switch, without routing packets to the edge. Certain VLANs, due to the requirements set by the PCI-DSS security standard, are isolated from other VLANs with specific access lists, allowing only the traffic that is passed to the core network from the edge network.

3.2. Virtual infrastructure automation

Virtual infrastructure automation can be applied to all aspects of the infrastructure. VMware vCenter API reveals all the same functionalities and methods that can be done with the graphical management utilities commonly used to manage the infrastructure. However, one of the most common use cases for such automation are virtualization host configuration and virtual machine provisioning.

3.2.1. Virtualization host configuration

Virtualization hosts were deployed by installing the currently latest available version of VMware ESXi. The management server, VMware vCenter, was brought into a single hypervisor as a pre-configured virtual appliance. All hypervisors were connected to the vCenter and they were added to a HA cluster. However, configuring many virtualization hosts manually was a time-consuming task. This was resolved by using VMware Host Profiles.

In order to implement a homogenous HA cluster, all hypervisors must be running with identical configuration. A host profile is an abstract configuration model of a hypervisor. When a non-configured hypervisor is brought into the environment the host can be configured by applying a host profile to it. For example, it is possible to define which datastores are added to the hypervisor, which TCP/IP heap size parameters are used and what networks are added to the host vSwitch.

A single virtualization host was configured manually. Based on that host, a host profile was created. The remaining hosts were configured by applying the host profile to them. By applying the host profile on all virtualization hosts, it is possible to check the host compliance with the host profile. This also enables to see if a host is misconfigured. If the host profile is changed somehow, it is also possible to remediate the profile to all virtualization hosts. This helps to make configuration changes on all hosts just by modifying the upstream host profile.

3.2.2. Virtual machine provisioning

Virtual machine provisioning was automated by creating a custom workflow with VMware vCenter Orchestrator. Virtual machines were provisioned from a virtual machine template, in this case a minimal installation of Red Hat Enterprise Linux 6 with 64-bit CPU architecture. The installation image was used as a base Linux virtual machine, which did not have any network interfaces or additional disks. All customization tasks were implemented in the workflow. The workflow is shown in Figure 3.2.

The workflow was designed to handle all tasks related to a virtual machine deployment through a single programmable interface. After deploying a new virtual machine from a template, the virtual machine itself can do very little. The virtual machine is powered

on, but the main problem is that the virtual machine is unable to configure its network configuration autonomously. Unattended network configuration can only be done via the hypervisor layer. With VMware Tools pre-installed to the virtual machine, the hypervisor can configure the network settings as specified in the workflow run. If the deployed virtual machine is configured to have multiple network interfaces, all of them can be configured with the workflow. All network interfaces are created with the type of VMXNET 3 to provide best network performance. This decision was based on VMware's best practices and benchmarks [45, 46].

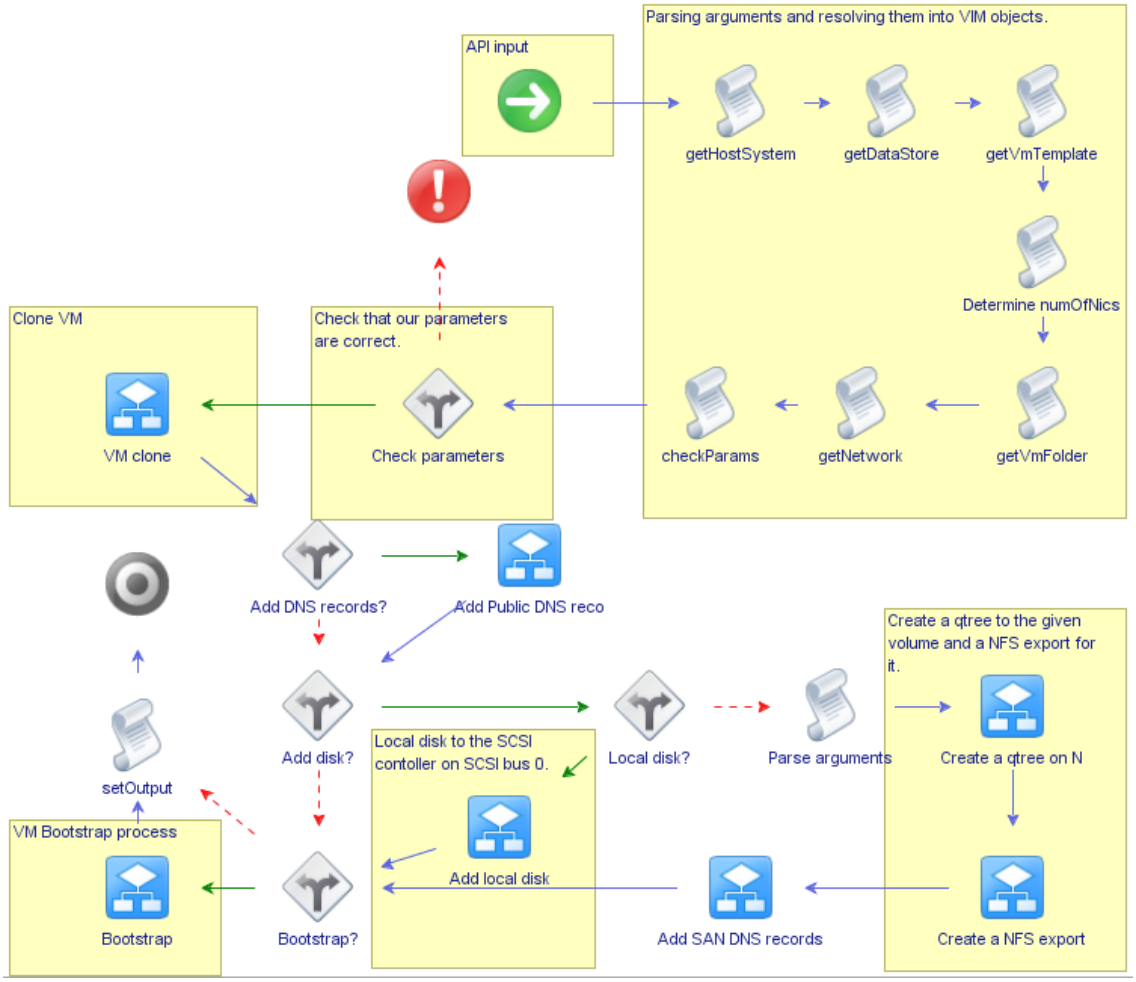

Figure 3.2: An Orchestrator workflow, implementing a virtual machine provisioning process in full. Input is given in an API call, which determines which parts of the workflow are executed.

After powering on the system and configuring the network, the storage is configured. If specified, additional disks are added to the virtual machine. Size and datastore are specified in the workflow input. Only a single additional disk can be added, but the workflow can be modified to support the addition of multiple disks to the system. The

first free SCSI controller number is chosen for the disk. The disk data is provisioned by using thin-provisioning, not reserving space, that has not been yet used.

If no additional disks are added to the system, additional storage can be provisioned on the storage system and then exported to the virtual machine. It is possible to define the storage controller and the volume. A storage qtree will be created to the given volume with the virtual machine hostname. This qtree can only be accessed with NFS protocol. Therefore, this sort of additional storage requires a network interface to the SAN network.

In addition to the actual provisioning, other services can be integrated to the workflow as well. For example, the workflow can create DNS records to the IP addresses configured to the machine. These DNS records are created on both private and public name servers, consisting of private and public DNS zones. SAN DNS zones are allocated from RFC 1918 private address space, and their DNS records should not be added to the public DNS servers [47].

3.2.3. API-based workflow control

As seen in Figure 3.2, the actual workflow run is started by giving input parameters via an API call. VMware vCenter Orchestrator API can be accessed by HTTPS protocol. The API supports GET and POST HTTP requests, which can be used to execute a workflow and follow its current state. The client interactions with the API are shown in Figure 3.3.

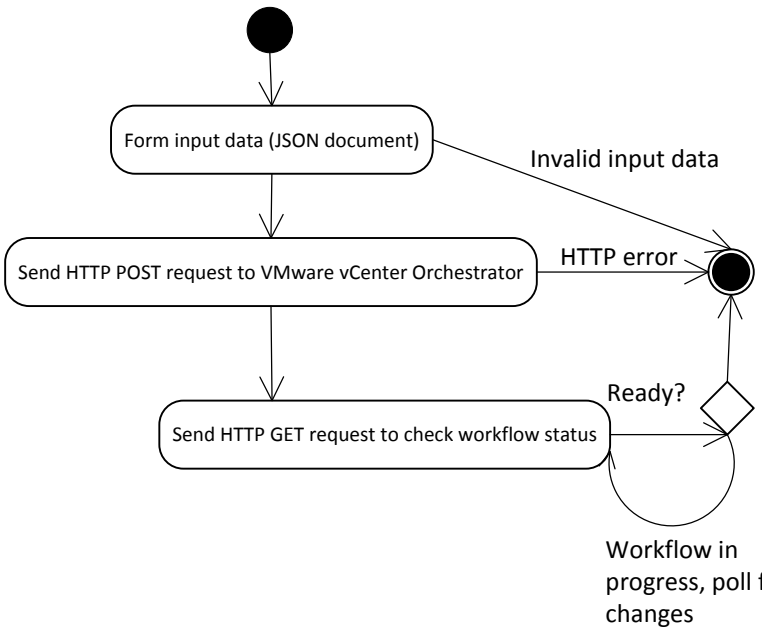

Figure 3.3: A state-chart showing the implementation of client-to-API interactions.

The workflow API takes the input parameters as HTTP post data in JSON or XML format. All Orchestrator workflows take the input parameters as a JSON array. A workflow, which does not have any input parameters, takes an empty JSON array as a paremeter. In order to make a complete API call, the input document must match workflow input parameters. The API supports multiple datatypes, such as strings, arrays and object references. If XML is used, the input document can be validated by using Orchestrator XML schema definition. JSON schema definition can be used to validate JSON input documents and datatypes.

The workflow was designed to take various input parameters, which allows as much customization as possible. These input parameters are presented on Table 3.1.

Table 3.1: Workflow API input parameters.

Input parameter name	**Data type**	**Description**
esxhost	string	ESXi host, which will host the provisioned virtual machine.
datastore	string	Datastore for virtual machine data.
vmfolder	string	vCenter inventory folder for provisioned virtual machine.
hostingLabel	string	Port group for virtual machine's primary network interface
inventoryName	string	vCenter inventory name for virtual machine.
hostname	string	Virtual machine hostname without the domain part.
dnsServers	array	An array of DNS servers.
dnsDomain	string	DNS domain for virtual machine hostname.
hostingGateway	string	Default gateway.
hostingIP	string	Primary IP address.
hostingNetmask	string	Netmask for the primary network interface.
powerOn	boolean	After provisioning the machine can be powered on.
numCPUs	integer	Number of virtual CPUs.

memoryMB	integer	Amount of allocated physical memory.
addInterface	boolean	Determines if an additional network interface is added.
additionalIP	string	IP address for additional network interface.
additionalNetmask	string	Netmask for additional network interface.
additionalLabel	string	Port group for additional network interface.
addPublicDNS	boolean	Determines if public DNS records are created.
runBootstrap	boolean	Determines if virtual machine is bootstrapped.
addDisk	boolean	Determines if an additional data disk is to be allocated.
diskType	string	Type of additional data disk. Can be a local disk or a storage qtree.
diskSource	string	Location of the data disk.
diskSize	integer	Size of the data disk in gigabytes.

The raw example JSON document containing all the input parameters can be found as an appendix listing A.1.

3.3. Resource and network allocation

Virtual machine resources can be controlled automatically. Table 3.1 shows how the workflow can be run with a parameter addDisk, which adds either a local disk to the system or allocates a qtree on a storage controller. As with other virtual machine related operations, the actual automation workflow was implemented with VMware vCenter Orchestrator. Storage controller operations were implemented with NetApp Data ONTAP API.

In addition to storage allocation, it is possible to control virtual machine CPU and memory allocation. By using CPU and memory hot-plug features, it is possible to increase

these resources on the fly. This sets requirements to the machine's virtual hardware, which is required to support hot-add features.

3.3.1. Storage automation

Qtree allocation was implemented in a Python application, which was integrated into the Orchestrator. Orchestrator itself does not support XML-RPC APIs natively, so a separate workflow was created for each major functionality. These include qtree allocation and storage export creation, which create necessary access control modifications to the newly created qtree. Separate workflows were created, as storage export creation can target qtrees that are already exported to a machine.

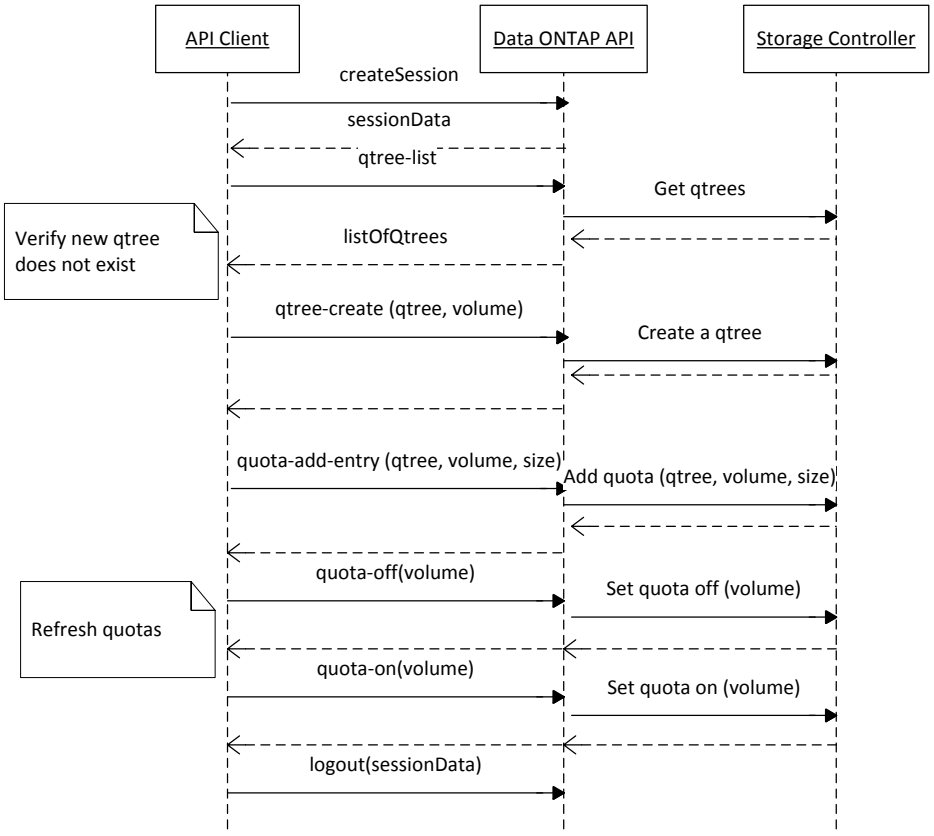

Figure 3.4: A sequence diagram showing API interactions between the client and the Data ONTAP API. API controls the actual storage controller.

3.3.2. CPU and memory allocation

The virtualization layer supports CPU and memory hot-add. This means that additional virtual CPUs and memory can be added to the system with certain limitations. The

guest operating system must support hot-add, and the guest virtual hardware must have CPU/memory hot-add feature enabled. If the guest CPU is running with 32-bit CPU architecture, memory cannot be increased over 4 GB due to limitations in the physical address space. Virtual CPUs can also be removed from the machine while it is running, but memory can only be increased.

All virtual machines were reviewed and the CPU and memory hot-add feature was enabled. A guest reboot was required, as the feature cannot be enabled when the machine is running. Guest virtual hardware was upgraded at the same time. This was done manually. No automation was created to modify these machine resources, as resources are modified infrequently and they can be easily increased with the virtualization layer management tools.

4. DISCUSSION

In this chapter, additional issues related to an SDDC are discussed. Possible improvements, such as environment upgrades, were not handled in the implementation chapter. Even though the infrastructure can be automated to provide automation tasks presented in Chapter 3, the operating system layer is still operated and configured without automation.

In addition to the automation presented in Chapter 3, additional features are discussed. In an enterprise environment, the automation can be used to provision pre-configured environments. Another important use case for such automation are different phases in software development, such as quality assurance and staging. This is discussed in this chapter as well.

4.1. Environment upgrades and security patches

The infrastructure described in Chapter 3 was designed to scale out well. The virtualization hosts were selected to have overly sized amount of RAM for scaling purposes. In addition to virtual machine memory scaling, it allows to move virtual machines within the environment from one host to another.

In case of a hardware upgrade, the host must be shutdown and before that all virtual machines must be migrated to another hosts. The total amount of used memory by virtualization hosts must be within a range that a single host can be cleared out of virtual machines. The selected hardware for virtualization hosts were chosen to have 128 GB of RAM. The memory setup will be upgraded to a maximum of 256 GB later.

The virtualization cluster supports scaling out as well. As described in Chapter 2, the HA cluster currently has a maximum of 32 hosts. An extra blade enclosure can be brought into the environment. As the infrastructure is based on components that can be changed within certain limitations, the components described in the reference architecture can be

replaced by different hardware, even from a different vendor.

4.2. IPv6 deployment

Internet Protocol version 4 (IPv4) provides 2^{32} IP addresses. Due to the growth of the Internet, the address space has moved to an exhausted state. Its successor, Internet Protocol version 6 (IPv6) provides 2^{128} IP addresses, but it is not compatible with IPv4. This sets challenges to an IT infrastructure, as both protocols must be supported in order to provide connectivity to all IP clients. However, the IPv6 adoption has still been low [48].

The infrastructure presented in Chapter 3 was deployed with only IPv4. IPv6 adoption was kept as an option for future developments, as the current need for IPv6 was low. The infrastructure components, consisting of storage, network and compute layers all support IPv6. This means that IPv6 adoption is only a configuration issue and does not require any hardware changes. The deployment can be done as a dual-stack, where all machines have both IPv4 and IPv6 connectivity.

Even though the infrastructure supports IPv6, the tools presented in Chapter 3 need to be modified to understand and support IPv6 addresses. The biggest change is the way IPv6 addresses are allocated to the machines in provisioning phase. One possible solution to this is to use RFC 2373 defined EUI-64 (Extended Unique Identifier) addresses, where the host itself can determine its own IPv6 address by using the interface MAC address [49]. The 48-bit MAC addresses are converted into 64-bit by splitting the MAC address to two 24-bit parts, inserting a 16-bit value 0xFFFE into the middle of the two, and then joining them. The virtualization layer guarantees that all virtual machines have a unique MAC address.

4.3. Automated configuration management

Usually after provisioning, a virtual machine must be configured for the task it was provisioned for. Automating the provisioning part of the process saves a lot of time and also eliminates human error. The same can be applied to system configuration. A part of the configuration can be set on the virtual machine template, for example certain kernel parameters and other generic configuration, that has to be same on all machines despite

its intended use. However, most of the configuration is machine specific. For example, storage NFS exports need to be mounted from the storage controller with a certain mount path. Server SNMP configuration, such as a unique SNMP community string needs to be generated for system monitoring purposes. Maintaining these configuration files manually within the machine is very straightforward, but modifying the configuration files and then propagating the same change to all machines is impossible. If, for example, a DNS server IP address is changed, the name resolving configuration must be modified on all machines. Manual work is error-prone and time-consuming.

A solution to this is centralized configuration management. This can be achieved with various tools, for example with Puppet [50]. Puppet is an automation architecture, which constists of one or more master nodes, including configuration manifests, which is propagated to clients. The arcitechture is described in Figure 4.1. The manifests are described by a domain-specific language. The actual Puppet software is implemented in Ruby. All clients periodically download the catalog from the master. All tasks described in the catalog are executed and enforced. If the manifest is changed since the last run, only the modified parts of the manifest is executed.

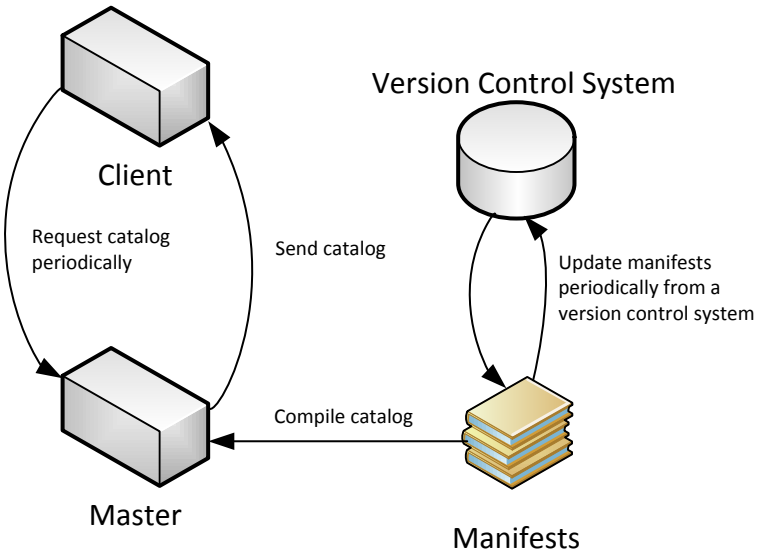

Figure 4.1: Puppet client requests catalogs periodically from a Puppet master, which compiles catalogs from manifests. These manifests are kept in a version control system.

In addition to configuration management, the software deployment can be automated as well. Software can be packaged to operating system specific packages, which can be installed with a software package manager. A software package only includes the necessary executables to run the software. All configuration is handled by the

configuration management tool. In this way, the software deployment can be even included in the provisioning phase. After virtual machine provisioning, the machine is powered on, as described in Chapter 3. When the operating system is running, all required software is installed. To this point the process can be always reproduced and repeated. This can be used to generate reproducible replicas of a software environment, such as staging. It is a prequisite for agile software development and continuous delivery.

4.4. IP address management

The environment described in Chapter 3 is based on static IP addresses. All machines have static IP configuration, which is configured automatically during machine provisioning. These IP addresses are managed externally outside of the virtualization environment. IP address management (IPAM) is used as a term to describe a software, that is used for managing, planning and tracking IP address usage [51].

As seen in Table 3.1, the workflow takes two IP addresses as an input. Currently this requires manual work, as the IP allocation has to be done before running the workflow. An improvement to this would be an IPAM integration, which would allocate an IP address automatically from the given network specified by input parameter hostingLabel. This integration can be implemented in the provisioning workflow. The IPAM software must have an API, which the orchestration tool can then utilize.

4.5. Implementing self-service provisioning

As presented in Chapter 3, Orchestrator workflows can be controlled with an API. It does not limit which kind of clients can use the API. A command-line utility is an easy way to implement a client. However, the API itself shows multiple different parameters, many of which are only useful to a system administrator. A developer, for example, is not interested in which virtualization host the virtual machine should be provisioned to. Another use case is a customer, who should only be able to create new virtual machines, and should only see minimal information about the environment. This is why a single command-line utility does not suffice.

In order to provide an easy-to-use interface to the provisioning system, a web interface can be created, which utilizes the Orchestrator API. A web interface can be implemented

to a variety of use cases, ranging from developers to customers. Depending on the use case, the web interface can show only the minimal amount of required input fields and information. In case of a developer, the web interface should only require a system hostname, which the developer can use to reach the machine. Another important feature is the configuration management presented in this chapter. This allows the developer to create a reproducible environment, usually similar to production, without interfering any other applications.

5. CONCLUSION

The efficiency and the reliability of server provisioning are becoming increasingly important. A key factor to achieve this is by utilizing virtualization technologies in all storage, networking and computing layers. This sets requirements both hardware and software-wise to the environment, which should be carefully selected. As the demand for resources grows, the environment must be scalable.

A software-defined datacenter is an architecture that is capable of resolving these problems today. It is a loosely defined term, which can be applied to a variety of vendors with various networking, computing and storage technologies. This thesis presents one possible design to this by utilizing VMware, NetApp and HP technologies. However, the solution is not entirely vendor-specific, and could be achieved by other technologies by adapting the implementation.

Network virtualization, or SDN, was presented but not fully utilized in the implementation. Virtualized switch was implemented with HP's proprietary protocol. OpenFlow protocol, supporting multi-vendor network configurations was not used. However, the hardware presented in Chapter 3 supports OpenFlow, and it could be applied to the environment in the future.

The implementation presented in Chapter 3 relies heavily on different APIs. API usage is not limited to the systems presented, and the workflow can be extended by integrating it to other systems. For example, the workflow could take a set of firewall rules as an input. These rules could then be propagated onto the firewalls located in the edge network.

A key improvement is the configuration management presented in Chapter 4. It is the next step in deployment automation, as the infrastructure automation is implemented. When both infrastructure and deployment is automated, the process can be facilitated significantly. Therefore, with certain limitations, the process could be executed by a non-administrator user, such as a developer.

The work presented in this thesis succeeded in its goals. The most time-consuming and

also the most used functionalities of the infrastructure were automated. The infrastructure presented was designed in a way, that also allows developing the infrastructure automation even further.

REFERENCES

[1] Adams, Keith and Agesen, Ole. A Comparison of Software and Hardware Techniques for x86 Virtualization, 2006, [WWW]. Accessed on 8.12.2013. Available at: http://www.vmware.com/pdf/asplos235_adams.pdf.

[2] Chang, Victor and Bacigalupo, David and Wills, Gary and De Roure, David. A Categorisation of Cloud Computing Business Models. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference*, pages 509–512. IEEE, 2010.

[3] Ernst & Young. Cloud Computing Issues and Impacts, 2011, [WWW]. Accessed on 7.12.2013. Available at: http://www.ey.com/Publication/vwLUAssets/Cloud-computing_issues_and_impacts/$FILE/Cloud_computing_issues_and_impacts.pdf.

[4] Zhang, Qi and Cheng Lu and Boutaba Raouf. Cloud Computing: State-of-the-art And Research Challenges. *Journal of Internet Services and Applications*, 01(01):7–18, May 2010.

[5] Humble, Jez and Molesky, Joanne. Why Enterprises Must Adopt Devops to Enable Continous Delivery. *Cutter IT Journal*, 24(8):6–13, August 2011.

[6] VMware. Delivering IT as a Service with a Software-Defined Data Center, 2012, [WWW]. Accessed on 8.12.2013. Available at: http://www.vmware.com/files/include/microsite/sddc/delivering-IT-as-a-service-with-a-SDDC.pdf.

[7] Marston, Sean and Marston, Sean and Bandyopadhyay, Subhajyoti and Zhang, Juheng and Ghalsasi, Anand. Cloud Computing - The Business Perspective. *Decision Support Systems*, 51(01):176–189, April 2011.

[8] EMC Corporation. An IT-as-a-Service Handbook: Ten Key Steps On The Journey to ITaaS, 2012, [WWW]. Accessed on 5.1.2014. Available at: http://www.emc.com/collateral/software/white-papers/h10801-stepstoitaas-wp.pdf.

[9] Dell Inc. and VMware Inc. How to Really Save Money with Virtualization, 2011, [WWW]. Accessed on 15.1.2014. Available at: http://www.vmware.com/files/pdf/partners/dell/dell-vmware_how-to-really-save-money.pdf.

[10] Rasmussen, Erik R. Reducing IT Costs and Increasing IT Efficiency by Integrating Platform-Virtualization in the Enterprise, 2009, [WWW]. Accessed on 21.4.2014. Available at: https://scholarsbank.uoregon.edu/xmlui/bitstream/handle/1794/8571/Rasmussen-2009.pdf.

[11] Amazon. Amazon Case Studies, [WWW]. Accessed on 6.5.2014. Available at: https://aws.amazon.com/solutions/case-studies/.

[12] Metzler, Jim. Riverbed Technology. Virtualization: Benefits, Challenges, and Solutions, 2011, [WWW]. Accessed on 15.1.2014. Available at: http://resources.idgenterprise.com/original/AST-0059219_Virtualization_-_Benefits_Challenges_and_Solutions.pdf.

[13] VMware Inc. VMware vCenter Server Datasheet, 2012, [WWW]. Accessed on 19.1.2014. Available at: http://www.vmware.com/files/pdf/products/vCenter/VMware-vCenter-Server-Datasheet.pdf.

[14] VMware Inc. VMware vSphere 5.1 vMotion Architecture, Performance and Best Practices, 2012, [WWW]. Accessed on 19.1.2014. Available at: http://www.vmware.com/files/pdf/techpaper/VMware-vSphere51-vMotion-Perf.pdf.

[15] VMware Inc. VMware High Availability Datasheet, 2009, [WWW]. Accessed on 19.1.2014. Available at: http://www.vmware.com/files/pdf/VMware-High-Availability-DS-EN.pdf.

[16] VMware Inc. VMware vSphere 5.5 Configuration Maximums, 2013, [WWW]. Accessed on 19.1.2014. Available at: http://www.vmware.com/pdf/vsphere5/r55/vsphere-55-configuration-maximums.pdf.

[17] VMware Inc. VMware vSphere High Availability 5.0 Deployment Best Practices, 2013, [WWW]. Accessed on 19.1.2014. Available at: http://www.vmware.com/files/pdf/techpaper/vmw-vsphere-high-availability.pdf.

[18] Peltz, Chris. Web services orchestration and choreography. *Computer*, 36:46–52, October 2003.

[19] VMware Inc. VMware vCenter Orchestrator, [WWW]. Accessed on 19.1.2014. Available at: http://www.vmware.com/products/vcenter-orchestrator/features.html.

[20] Info-Tech Research Group. Vendor Landscape: Mid-range to Entry Enterprise Storage Arrays, 2012, [WWW]. Accessed on 27.3.2014. Available at: http://media.netapp.com/documents/ar-mid-range-to-entry-enterprise-storage.pdf.

[21] NetApp. Slash Your Data Center Costs Today with NetApp Consolidation Solutions, 2009, [WWW]. Accessed on 27.3.2014. Available at: http://media.netapp.com/documents/ds-2892.pdf.

[22] Baltazar, Henry. Software-Defined Storage Will Sound The Death Knell For Traditional Storage Provisioning, 2013, [WWW]. Accessed on 27.3.2014. Available at: http://media.netapp.com/documents/wp-software-defined-storage-forrester.pdf.

[23] NetApp. NetApp Clustered Data ONTAP 8.2: An Introduction, 2013, [WWW]. Accessed on 29.3.2014. Available at: http://media.netapp.com/documents/tr-3982.pdf.

[24] NetApp. How HA pairs relate to the cluster, 2013, [WWW]. Accessed on 29.3.2014. Available at: https://library.netapp.com/ecmdocs/ECMP1196905/html/GUID-A8DBB1CE-ADEB-4C44-91AB-F1BD0120E77B.html.

[25] White, Jay and Lueth, Chris and Bell, Jonathan. NetApp. RAID-DP: NetApp Implementation of Double-Parity RAID for Data Protection, 2013, [WWW]. Accessed on 29.3.2014. Available at: http://media.netapp.com/documents/tr-3298.pdf.

[26] Kang, Woon-Hak and Lee, Sang-Won and Moon, Bongki. Flash-based extended cache for higher throughput and faster recovery. *VLDB Endowment*, 5(11):1615–1626, July 2012.

[27] NetApp. NetApp Flash Cache Datasheet, 2013, [WWW]. Accessed on 29.3.2014. Available at: http://media.netapp.com/documents/ds-2811.pdf.

[28] NetApp. NetApp Manageability SDK, 2014, [WWW]. Accessed on 29.3.2014. Available at: https://communities.netapp.com/docs/DOC-1152.

[29] IEEE Standards Association. IEEE Standard for Local and metropolitan area networks - Link Aggregation, 2008, IEEE 802.1AX-2008. Available at: http://standards.ieee.org/getieee802/download/802.1AX-2008.pdf.

[30] Hewlett-Packard. Intelligent Management Center, [WWW]. Accessed on 9.5.2014. Available at: www.hp.com/networking/IMC-eAPI.

[31] Open Networking Foundation. Software-Defined Networking: The New Norm for Networks, 2012, [WWW]. Accessed on 29.3.2014. Available at: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf.

[32] Cisco. Cisco Application Centric Infrastructure, 2013, [WWW]. Accessed on 20.4.2014. Available at: http://www.cisco.com/c/dam/en/us/products/collateral/cloud-systems-management/aci-fabric-controller/at-a-glance-c45-729864.pdf.

[33] European Telecommunications Standards Institute. Network Functions Virtualisation. An Introduction, Benefits, Enablers, Challenges & Call for Action, 2012, [WWW]. Accessed on 20.4.2014. Available at: http://portal.etsi.org/NFV/NFV_White_Paper.pdf.

[34] Hewlett-Packard. Reducing network complexity, boosting performance with HP IRF technology, 2010, [WWW]. Accessed on 30.3.2014. Available at: http://h17007.www1.hp.com/docs/reports/irf.pdf.

[35] Cisco. Routing Between VLANs Overview, [WWW]. Accessed on 30.3.2014. Available at: http://www.cisco.com/c/en/us/td/docs/ios/12_2/switch/configuration/guide/fswtch_c/xcfvl.pdf.

[36] VMware. vSphere Networking, 2013, [WWW]. Accessed on 30.3.2014. Available at: http://pubs.vmware.com/vsphere-55/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-55-networking-guide.pdf.

[37] Cisco. IPsec VPN WAN Design Overview, 2007, [WWW]. Accessed on 30.3.2014. Available at: http://www.cisco.com/application/pdf/en/us/guest/netsol/ns171/c649/ccmigration_09186a008074f22f.pdf.

[38] Mell, Peter and Scarfone, Karen. Guide to Intrusion Detection and Prevention Systems (IDPS). Technical report, National Institute of Standards and Technology, U.S. Department of Commerce, February 2007. Special Publication 800-94.

[39] VMware. VMware Distributed Switch, [WWW]. Accessed on 30.3.2014. Available at: http://www.vmware.com/products/vsphere/features/distributed-switch.html.

[40] International Organization for Standardization. ISO/IEC 27001 - Information security management, 2013. Available at: http://www.iso.org/iso/home/standards/management-standards/iso27001.htm.

[41] International Organization for Standardization. ISO 9000 - Quality management, 2005. Available at: http://www.iso.org/iso/home/standards/management-standards/iso_9000.htm.

[42] International Organization for Standardization. ISO 14000 - Environmental management, 2004. Available at: http://www.iso.org/iso/home/standards/management-standards/iso14000.htm.

[43] PCI Security Standards Council. Payment Card Industry (PCI) Data Security Standard - Requirements and Security Assessment Procedures, 2010, [WWW]. Accessed on 30.3.2014. Available at: https://www.pcisecuritystandards.org/documents/pci_dss_v2.pdf.

[44] PCI Security Standards Council. Why Comply with PCI Security Standards?, [WWW]. Accessed on 30.3.2014. Available at: https://www.pcisecuritystandards.org/security_standards/why_comply.php.

[45] VMware. Performance Evaluation of VMXNET3 Virtual Network Device, [WWW]. Accessed on 7.5.2014. Available at: http://www.vmware.com/pdf/vsp_4_vmxnet3_perf.pdf.

[46] VMware. Choosing a network adapter for your virtual machine, [WWW]. Accessed on 7.5.2014. Available at: http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1001805.

[47] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address Allocation for Private Internets. RFC 1918. IETF, 1996. Available at: http://tools.ietf.org/html/rfc1918.

[48] Collitti, Lorenzo and H. Gunderson, Steinar and Kline, Erik and Refice, Tiziana. Evaluating IPv6 Adoption in the Internet. *Passive and Active Measurement*, 6032:141–150, April 2010.

[49] B. Hinden, S. Deering. IP Version 6 Addressing Architecture. RFC 2373. IETF, 1998. Available at: http://tools.ietf.org/html/rfc2373.

[50] Puppet Labs. What is Puppet?, [WWW]. Accessed on 15.4.2014. Available at: http://puppetlabs.com/puppet/what-is-puppet.

[51] TechTarget. IP address management (IPAM) definition, [WWW]. Accessed on 21.4.2014. Available at: http://whatis.techtarget.com/definition/IP-address-management-IPAM.

A. ORCHESTRATOR API EXAMPLE JSON DOCUMENT

Listing A.1: A raw JSON document which holds example data.

```
{
    "parameters": [
        {
            "name": "esxhost",
            "scope": "local",
            "type": "string",
            "value": {
                "string": {
                    "value": "esxhost01.vsphere.local"
                }
            }
        },
        {
            "name": "datastore",
            "scope": "local",
            "type": "string",
            "value": {
                "string": {
                    "value": "netapp01-head-a.netapp.local,vm01"
                }
            }
        },
        {
            "name": "vmfolder",
            "scope": "local",
            "type": "string",
            "value": {
                "string": {
                    "value": "Virtual Machines"
                }
            }
        },
        {
            "name": "hostingLabel",
            "scope": "local",
            "type": "string",
            "value": {
                "string": {
                    "value": "Hosting Network"
                }
            }
        },
        {
            "name": "inventoryName",
            "scope": "local",
            "type": "string",
            "value": {
                "string": {
```

```
                    "value": "RHEL6_64-vco-example-vm.example.org"
                }
            }
        },
        {
            "name": "hostname",
            "scope": "local",
            "type": "string",
            "value": {
                "string": {
                    "value": "vco-example-vm"
                }
            }
        },
        {
            "name": "dnsServers",
            "scope": "local",
            "type": "Array/string",
            "value": {
                "array": {
                    "elements": [
                        {
                            "string": {
                                "value": "8.8.8.8"
                            }
                        },
                        {
                            "string": {
                                "value": "8.8.4.4"
                            }
                        }
                    ]
                }
            }
        },
        {
            "name": "dnsDomain",
            "scope": "local",
            "type": "string",
            "value": {
                "string": {
                    "value": "example.org"
                }
            }
        },
        {
            "name": "hostingGateway",
            "scope": "local",
            "type": "Array/string",
            "value": {
                "array": {
                    "elements": [
                        {
                            "string": {
                                "value": "10.0.0.1"
                            }
                        }
                    ]
                }
            }
        },
        {
            "name": "hostingIP",
            "scope": "local",
```

```
            "type": "string",
            "value": {
                "string": {
                    "value": "10.0.0.100"
                }
            }
        },
        {
            "name": "hostingNetmask",
            "scope": "local",
            "type": "string",
            "value": {
                "string": {
                    "value": "255.255.255.0"
                }
            }
        },
        {
            "name": "powerOn",
            "scope": "local",
            "type": "boolean",
            "value": {
                "boolean": {
                    "value": true
                }
            }
        },
        {
            "name": "numCPUs",
            "scope": "local",
            "type": "number",
            "value": {
                "number": {
                    "value": 4.0
                }
            }
        },
        {
            "name": "memoryMB",
            "scope": "local",
            "type": "number",
            "value": {
                "number": {
                    "value": 8192.0
                }
            }
        },
        {
            "name": "addInterface",
            "scope": "local",
            "type": "boolean",
            "value": {
                "number": {
                    "value": true
                }
            }
        },
        {
            "name": "additionalIP",
            "scope": "local",
            "type": "string",
            "value": {
                "string": {
                    "value": "10.100.0.100"
```

```
                }
            }
        },
        {
            "name": "additionalNetmask",
            "scope": "local",
            "type": "string",
            "value": {
                "string": {
                    "value": "255.255.255.0"
                }
            }
        },
        {
            "name": "additionalLabel",
            "scope": "local",
            "type": "string",
            "value": {
                "string": {
                    "value": "SAN Network"
                }
            }
        },
        {
            "name": "addPublicDNS",
            "scope": "local",
            "type": "boolean",
            "value": {
                "boolean": {
                    "value": true
                }
            }
        },
        {
            "name": "runBootstrap",
            "scope": "local",
            "type": "boolean",
            "value": {
                "boolean": {
                    "value": true
                }
            }
        },
        {
            "name": "addDisk",
            "scope": "local",
            "type": "boolean",
            "value": {
                "boolean": {
                    "value": true
                }
            }
        },
        {
            "name": "diskType",
            "scope": "local",
            "type": "string",
            "value": {
                "string": {
                    "value": "SAN"
                }
            }
        },
        {
```

```
            "name": "diskSource",
            "scope": "local",
            "type": "string",
            "value": {
                "string": {
                    "value": "netapp01-head-a.netapp.local,nfsStorage"
                }
            }
        },
        {
            "name": "diskSize",
            "scope": "local",
            "type": "number",
            "value": {
                "number": {
                    "value": 20.0
                }
            }
        }
    ]
}
```